УДК: 004.932.2

МОДИФИКАЦИЯ СТАТИСТИЧЕСКОГО АЛГОРИТМА ВЫДЕЛЕНИЯ КОНТУРОВ

Кириченко Ю. В.

Национальный технический университет Украины "Киевский политехнический институт имени Игоря Сикорского", Украина, Киев

Целью данной работы было создание, реализация и сравнение модифицированного алгоритма выделения контура, который базируется на статистическом алгоритме, с классическим статистическим алгоритмом. В результате проведенной работы был модифицирован алгоритм, описаны основные особенности, проблемы и требования к данному семейству алгоритмов и особенности реализации самого алгоритма. Также написан программный код на языке программирования Java, при исполнении которого осуществляется выделение контуров на изображении двумя способами: классическим и модифицированным. Также были протестированы возможности, сложность, время выполения данных алгоритмов. Результаты данной работы могут быть использованы в любых системах распознавания образов. Также были рассмотрены недостатки подобных алгоритмов и сформированы цели для дальнейшего исследования этой темы.

Ключевые слова: выделение конутров, статистическое контурирование, разпознавание образов, алгоритмы выделения контуров.

Кириченко Ю. В. Модифікація статистичного алгоритму виділення контурів / Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського", Україна, київ.

Метою даної роботи було створення, реалізація і порівняння модифікованого алгоритму виділення контуру, який базується на статистичному алгоритмі, з класичним статистичним алгоритмом. В результаті проведеної роботи був модифікований алгоритм, описані основні особливості, проблеми і вимоги до даного сімействі алгоритмів і особливості реалізації самого алгоритму. Також написаний програмний код на мові програмування Java, при виконанні якого здійснюється виділення контурів на зображенні двома способами: класичним і модифікованим. Протестовані можливості, складність, час виконання даних алгоритмів. Результати даної роботи можуть бути використані в будь-яких системах розпізнавання образів. Також були розглянуті недоліки подібних алгоритмів і сформовані цілі для подальшого дослідження цієї теми.

Ключові слова: виділення контурів, статистичне конкурування, розпізнавання образів, алгоритми виділення контурів.

Y. V. Kyrychenko Modification of the statistical contouring algorithm / National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Ukraine, Kyiv.

The purposes of this work are to create, implement and compare a modified algorithm for contour detection which is based on a statisticalalgorithm with a classic statistic algorithm. As a result of the work, the algorithm was modified, described the main features of the problem, the requirements to contour detection algorithms and the problems of algorithm implementation. Also program code in the Java programming language is written, which detects contour on the image in two ways, classical and modified. Also was tested: features, complexity, time of execution of these algorithms. The results of this work can be used in any image recognition system. Also, the shortcomings of such

algorithms were considered and goals for further researches of this topic were formed.

Key words: contour detection, statistical contouring, image recognition, algorithms for contour detection.

Введение. В настоящее время существует тенденция автоматизации различных процессов. Немаловажную роль в этом отыгрывает распознавание образов, которое использует выделение контуров, распознаваемого обекта и во многом влияет на качество и скорость работы всей системы в целом.

Наиболее популярными методами выделения контуров являются:

- Выделение границ Кэнни;
- Оператор Прюитт;
- Перекрёстный оператор Робертса;
- Оператор Марра-Хилдрета;
- Анализ нейронными сетями;
- Статестический подход;
- Подходы, основанные на согласованности фаз[1].

У каждого из перечислиных методов есть как положительные, так и отрицательные стороны и они ориентированы на свою специфическую область применения. Например, подходы, основанные на согласованности фаз, имеют самую высокую точность выделения контуров, но сложны в реализации и требуют большие вычислительные затраты[2].

В данной статье предложена модификация алгоритма статистического подхода улучшения точности выделения контура и произведено сравнение его с классическим вариантом по критериям

четкость выделенных границ, правильность выделенных границ и время выполнения.

Алгоритмы выделения контуров будут реализованы и протестированы на языке программирования Java для вычисления на CPU.

Для реализации этих алгоритмов не будут использоваться сторонние библиотеки, чтобы избежать жесткой привязки к языку программирования и большей наглядности.

Классический алгоритм статистического выделения контуров производит:

- выбор рабочего окна;
- расчет среднего значения в окне по формуле

$$q = \frac{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} a_{ij}}{mn}; \quad (1)$$

где q – среднее значение в окне,

aij – элемент окна с номером ij,

m, n – размерность окна по горизонтали и вертикали соответственно.

- расчет среднеквадратического отклонения в окне по формуле

$$s = \sqrt{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left(a_{ij} - q\right)^2}; \quad (2)$$

- умножение среднеквадратического отклонения в окне на каждый элемент окна по формуле

$$a_{ij} = a_{ij} s \ . \quad (3)$$

Исходя из описанных формул, можно вычислить количество необходимых математических операций для выполнения алгоритма[1]. Примем количество пикселей изображения равным p, а количество пикселей окна за w, тогда количество операций для

второго шага равно p+p/w, для третьего 2p + p/w и для четвертого p. Исходя из этого общее количество элементарных мат. операций равно 4p+2p/w, так как большинство быстрых алгоритмов базируется на поэлементном умножении и суммировании всех элементов окна размерности[3] минимум 2x2(причем при случае 2x2 используется 2 прохода с разными матрицами и деление для каждого элемента полученных матриц) минимальное количество мат. операций равно 2(4p+3p)+p=15p. Отрицательной чертой алгоритма можно назвать размытость границ, низкую эффективность при нахождении границ в областях, где значение яркости близко к нулю. Для решения проблем алгоритма была разработана его модификация.

Модификация алгоритма основана на причинах проблем классического алгоритма. Причиной размытости рамок является анализ изображения в окне, не в зависимости от результатов вычислений соседних окон[4], что приводит к созданию визуальных квадратов областей. Проблему размытости можно решить, сделав дополнительный проход по изображению со сдвигом в половину размера окна по ширине и долготе изображения, изменив значения выходных элементов на максимальное из значений в ячейках матриц. Причиной низкой эффективности при нахождении границ областей низкой яркости является то, что алгоритм оценивает резкие перепады в области количественно[5], и при низких значениях относительно большой перепад для области будет абсолютно незначимым для изображения в целом. Данная проблема может быть решена с помощью нормирования каждого окна в определённом диапазоне с помощью формулы 4:

$$a_{ij} = a_{ij} / max(a), \quad (4)$$

где max(a) – максимальный элемент в окне.

Также для увеличения производительности нормирование стоит проводить в последнюю очередь. Оценка количества арифметических операций для одного писеля после оптимизации 2(4p+2p/w)+(p)+(2p)=11p +2p/w. Где: 4p+2p/w – количество мат. операций классического алгоритма, 2(4p+2p/w) – 2 прохода: обычный и со сдвигом, (p) – нахождение максимума для каждой точки, (2p) – нормализация по окну.

Для лучшего понимания работы реализации на Java приведем используемые методы классов и объясним их предназначение:

- Matrix – класс, инкапсулирующий в себе базовую работу с матрицей:
  - Метод set(int x,int y,int val) – устанавливает значение val в x-ую строку и y-тый столбец матрицы;
  - setMatrixIn (Matrix matrix,int x,int y) – заменяет значение в данной матрице значениями из matrix, начиная со строки x и столбца y;
  - getSubMatrix (int startX,int endX,int startY,int endY) – получение матрицы, которая является подмножеством данной матрицы, начиная с startX строки, startY столбца и заканчивая endX строкой и endY столбцом.
- MatrixMath – класс, инкапсулирующий в себе реализацию математических операций над матрицами:
  - mul(Matrix matrix,double val) – метод выполняет умножение матрицы на константу;
  - normalize(Matrix matrix, int max) – метод выполняет нормирование значений в матрице в диапазоне от 0 до max;

- squareDiffRight (Matrix matrix) – метод выполняет вычисление среднеквадратического отклонения для матрицы matrix;
- setToMax (Matrix matrixA,Matrix matrixB) – метод для нахождения максимума для каждого элемента матрицы из двух матриц;
- setToMin (Matrix matrixA,Matrix matrixB) – метод для нахождения минимума для каждого элемента матрицы из двух матриц.

- ImageUtils – класс инкапсулирует в себе обработку изображений:
  - showImg(Matrix matrix, Boolean invers) – отображает матрицу как изображение в черно-белых тонах, инвертируя его при invers true;
  - getColorPartArray(Matrix img,int part) получает определённый слой изображения (используется только BRIGHTNESS_PART – ярость).

Приложение имеет достаточно простую структуру – два метода, реализующие классический, модифицированный алгоритмы и точку входа в приложение, которая состоит из следующих частей: загрузка изображений, подготовка к основной части, оценка времени выполнения классического алгоритма, оценка времени выполнения модифицированного алгоритма и визуализация работы алгоритмов с последним изображением из списка. Данный класс показан в листинге 1.

Листинг 1

```
public class test {
	public static void main(String[] args){
		try {
			//загрузка изображений
			String path=new File(".").getAbsolutePath()+"\\autos\\";
```

```
                BufferedImage[] sampleImgs=new BufferedImage[8];
                for (int i = 0; i < sampleImgs.length; i++) {
                        sampleImgs[i]=ImageUtils.load(path+(i+1)+".jpg");
                }
                //подготовка к основной части
                int window=5;
                long start;
                Matrix classic = null;
                Matrix modif = null;
                //оценка времени выполнения классического
алгоритма
                start=System.currentTimeMillis();
                for (int i = 0; i < sampleImgs.length; i++) {
                        classic = statisticalTransClassic(window,
AlgoUtils.getColorPartArray(AlgoUtils.imgToMatrix(sampleImgs[i]),
ImageUtils.BRIGHTNESS_PART), 0);
                }
                System.out.println("среднее время выполнения
классического алгоритма "+((System.currentTimeMillis()-
start)/sampleImgs.length));
                //оценка времени выполнения модифицированного
алгоритма
                start=System.currentTimeMillis();
                for (int i = 0; i < sampleImgs.length; i++) {
                        Matrix modifiedPart1 = statisticalTransMod(window,
ImageUtils.getColorPartArray(AlgoUtils.imgToMatrix(sampleImgs[i]),
ImageUtils.BRIGHTNESS_PART), window/2);
                        Matrix modifiedPart2 = statisticalTransMod(window,
ImageUtils.getColorPartArray(AlgoUtils.imgToMatrix(sampleImgs[i]),
ImageUtils.BRIGHTNESS_PART), window/2);
                        modif=MatrixMath.setToMax(modifiedPart1,
modifiedPart2, (a,b)->true);
                }
                System.out.println("среднее время выполнения
модифицированного алгоритма "+((System.currentTimeMillis()-
start)/sampleImgs.length));
                //визуализация работы алгоритмов с последним
изображением из списка
                ImageUtils.showImg(sampleImgs[sampleImgs.length-1]);
                ImageUtils.showImg(modif, true);
                ImageUtils.showImg(MatrixMath.normalize(classic,
255),false);
            }catch (Exception e) {
                e.printStackTrace();
```

```
                    System.err.println("ошибка");
                }
        }
        //классический метод
        public static Matrix statisticalTransClassic(int window,Matrix
img,int ofset){
                Matrix res=new
Matrix(img.getRowLenhgth(),img.getColLength());
                for (int i = 0; i < (img.getRowLenhgth()-ofset)/window; i++) {
                    for (int j = 0; j < (img.getColLength()-ofset)/window; j++)
{
                            Matrix
windowMatrix=img.getSubMatrix(ofset+i*window,ofset+ (i+1)*window,
ofset+j*window,ofset+ (j+1)*window);
                            double
squareDiff=MatrixMath.squareDiffRight(windowMatrix);
                            Matrix
resultWindowMatrix=MatrixMath.mul(windowMatrix, squareDiff);
                            res.setMatrixIn(resultWindowMatrix, i*window,
j*window);
                    }
                }
                return res;
        }
        //модифицированный метод
        public static Matrix statisticalTransMod(int window,Matrix img,int
ofset){
        Matrix res=new Matrix(img.getRowLenhgth(),img.getColLength());
        for (int i = 0; i < (img.getRowLenhgth()-ofset)/window; i++) {
                for (int j = 0; j < (img.getColLength()-ofset)/window; j++) {
                    Matrix opMatrix=img.getSubMatrix(ofset+i*window,ofset+
(i+1)*window, ofset+j*window,ofset+ (j+1)*window);
                    double
squareDIff=MatrixMath.squareDiffRight(opMatrix);
                    Matrix
normalized=MatrixMath.normalize(MatrixMath.mul(opMatrix,
squareDIff),255);
                    res.setMatrixIn(normalized, i*window, j*window);
                }
        }
        return res;
}
}
```

В результате выполнения кода в листинге 1 были получены данные о разнице в скорости работы и получены 2 изображения. Скорость работы классического алгоритма в среднем составила 34 мс, а модифицированного – 66 мс, эти данные говорят о том, что модифицированный алгоритм при данной реализации примерно в 2 раза медленнее классического. Также количество элементарных мат. Операций на пиксель для классического алгоритма 4p+2p/w, а модифицированного 11p что в $\frac{11p}{4p+2p/w} \approx 2.8$ раза больше. Из рисунка 1 видно, что новый алгоритм (черный на белом фоне) выделяет контуры более точно, чем классический алгоритм (белый на черном фоне), также выделенные контуры не являются размытыми, как в классическом варианте.

Рис. 1 Результаты работы программы

Выводы. В данной работе был предложен алгоритм выделения контуров, основанный на классическом алгоритме статистического анализа. Также была представлена его реализация на языке программирования Java.

Были проанализированы недостатки классического алгоритма и предложены способы их решения в новом алгоритме. Также были проведены сравнительные тесты для определения скорости выполнения классического алгоритма и модифицированного.

Негативной стороной модифицированного алгоритма оказалась меньшая скорость, чем у классического, в 2 раза.

Дальнейшие исследования могут быть направлены на устранение выше изложенных недостатков, а также над модификацией алгоритма/реализации для использования на GPU.

Литература:

1. *V. Lacroix. “The Primary Raster: A Multiresolution Image Description”. In Proceedings of the 10th International Conference on Pattern Recognition, 1990.*

2. J. F.Canny. “A Computational Approach to Edge Detection”. IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6), Nov 1986.

3. D. Ziou and S. Tabbone. “A Multi-Scale Edge Detector”. Pattern Recognition, 26(9), 1993.

4. R. C. Gonzalez, R. E. Woods, Digital Image Processing, Prentice-Hall, Inc, Upper Saddle River, New Jersey, 2002.

5. S. Beucher, F. Meyer, The Morphological Approach to Segmentation: The Watershed Transformation, in “Mathematical Morphology in Image Processing”, E. R. Dougherty Editor, Marcel Dekker, Inc, New York 1992.

References:

1. V. Lacroix. “The Primary Raster: A Multiresolution Image Description”. In Proceedings of the 10th International Conference on Pattern Recognition, 1990.

2. J. F.Canny. “A Computational Approach to Edge Detection”. IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6), Nov 1986.

3. D. Ziou and S. Tabbone. “A Multi-Scale Edge Detector”. Pattern Recognition, 26(9), 1993.

4. R. C. Gonzalez, R. E. Woods, Digital Image Processing, Prentice-Hall, Inc, Upper Saddle River, New Jersey, 2002.

5. S. Beucher, F. Meyer, The Morphological Approach to Segmentation: The Watershed Transformation, in “Mathematical Morphology in Image Processing”, E. R. Dougherty Editor, Marcel Dekker, Inc, New York, 1992.